INFO-F-105

Projet C++

Année académique 2014–2015

Définition du problème

On souhaite manipuler des sous-ensembles de $\mathbb{R}$ définis par un nombre fini d'intervalles fermés dont les bornes sont des entiers. Par exemple, l'ensemble défini par

$$[-101, 2] \cup [15, 15] \cup [42, 1024].$$

Pour représenter de tels unions d'intervalles, nous vous demandons de créer un ADT `Uinter` qui sera une liste (simplement liée) d'intervalles. Cet ADT comportera comme attribut un pointeur `_tete` vers un objet de type `_Inter` (cette nouvelle classe qui représente les intervalles sera donc imbriquée dans `Uinter` et privée). Voici le squelette de `Uinter` :

```
class Uinter {
   class _Inter {
      int _bi, _bs;
      _Inter *_next;
   public:
      _Inter(int, int, _Inter*);
   };
   _Inter *_tete;
public:
   Uinter(): _tete(nullptr) {}
   void reunion(int, int);
   void printUinter();
   bool contient(int);
};
```

Nous avons dans la classe `_Inter` que `_bi` est destiné à stocker la borne inférieure de l'intervalle, `_bs` est destiné à stocker la borne supérieure de l'intervalle, et `_next` est un pointeur vers un autre intervalle. Attention, il faudra empêcher la création d'intervalle mal défini où la borne inférieure est strictement supérieure à la borne supérieure.

Il faudra bien entendu encore ajouter des getters/setters nécessaires.

Les listes d'intervalles doivent satisfaire les contraintes suivante :

- les intervalles considérés dans un même ensemble doivent avoir une intersection vide,

- les intervalles sont triés dans la liste en ordre croissant de leur borne inférieure.

Nous dirons qu'une liste qui satisfait ces contraintes est *canonique*.

Par exemple $[1,5],[4,7]$ n'est pas canonique, vu que les deux intervalles ont une intersection non vide (la liste canonique représentant cet ensemble est simplement constituée de l'intervalle $[1,7]$), pour les même raison $[0,1],[1,9]$ n'est pas non plus canonique. La liste $[10,12],[0,7]$ n'est pas canonique non plus, vu que le premier intervalle a une plus petite borne que le second.

Nous vous demandons de réaliser les trois méthodes suivantes pour la classe `Uinter` :

```
void reunion(int bi, int bs)
void printUinter()
bool contient(int nb)
```

La première méthode fait l'union entre l'ensemble représenté par la liste pointé par `_tete` et un nouvel intervalle dont les bornes sont `bi` et `bs`. La liste résultat doit être canonique et l'adresse de sa tête est placé dans `_tete`. Une autre façon de voir les choses est de considérer que la fonction «insère» l'intervalle $[bi,bs]$ dans la liste. La deuxième méthode imprimera à l'écran l'ensemble désigné par `_tete`. Enfin la troisième méthode renvoie True ou False si le nombre `nb` est ou n'est pas dans l'ensemble.

Illustration

Vous trouverez ci-dessous une série d'exemples d'applications de la méthode `reunion`. La première colonne donne la liste de départ, la deuxième l'intervalle avec lequel on fait l'union et la troisième le résultat attendu.

ensemble	intervalle	résultat
$[5,8][10,15][22,30]$	$[12,23]$	$[5,8][10,30]$
$[5,8][10,30]$	$[0,0]$	$[0,0][5,8][10,30]$
$[0,0][5,8][10,30]$	$[10,25]$	$[0,0][5,8][10,30]$
$[0,0][5,8][10,30]$	$[7,7]$	$[0,0][5,8][10,30]$
$[0,0][5,8][10,30]$	$[7,9]$	$[0,0][5,9][10,30]$
$[0,0][5,9][10,30]$	$[3,5]$	$[0,0][3,9][10,30]$
$[0,0][3,9][10,30]$	$[100,150]$	$[0,0][3,9][10,30][100,150]$
$[0,0][3,9][10,30][100,150]$	$[140,170]$	$[0,0][3,9][10,30][100,170]$
$[0,0][3,9][10,30][100,170]$	$[-1,0]$	$[-1,0][3,9][10,30][100,170]$
$[-1,0][3,9][10,30][100,170]$	$[-100,1000]$	$[-100,1000]$

Indications complémentaires

Pour réaliser la méthode `reunion(int bi, int bs)` demandée, nous vous conseillons de trouver tous les intervalles dans la liste désignée par `_tete` qui ont une intersection avec $[bi,bs]$ et remplacer le tout par un seul intervalle dont la borne inférieure est le minimum de *bi* et des bornes inférieures des intervalles trouvés et dont la borne supérieure est le maximum de *bs* et des bornes supérieures des intervalles trouvés.

Vous êtes libres de rajouter des méthodes au squelette tant de `_Inter` que de `Uinter` si cela est pertinent. Attention dans votre implémentation à bien libérer la mémoire de tous les intervalles devenus inutilisés.

Consignes pour la remise du projet

À respecter scrupuleusement !

1. Votre projet doit comporter **votre nom** et **votre section**.
2. Votre projet doit être **dactylographié**. Les projets écrits à la main ne seront **pas corrigés**.
3. Votre code doit être **commenté**.
4. Vous devez respectez les modalités suivantes :
   — Rendre une copie papier du projet au secrétariat étudiant.
   — Poster vos fichiers source sur l'UV.
   — Date : **le 24 mars**
   — Heure : **Avant 10h**

   Après 10h, les projets sont considérés comme **en retard**, et vous perdez **1 point** sur votre note finale (plus un point par jour de retard).